DURCHS WILDE POKEISTAN

Ein Wegweiser durch den Speicherdschungel

Zum befriedigenden Programmieren sind Peek und Poke unverzichtbares Werkzeug. Um es sinnvoll einzusetzen, sind Grundkenntnisse nötig, von denen hier die Rede sein wird.

Der Commodore 64 hat zwar eine Menge leistungsfähiger Hardware eingebaut, wie zum Beispiel den VIC-II-Videochip oder SID, den Musikchip, aber leider fehlen ihm die Befehle, diese Fähigkeiten angemessen auszunutzen und dabei noch einfach zu programmieren. Natürlich kann man mit Befehlserweiterungen arbeiten. Zum einen ist es aber sehr langwierig, vor jeder Programmänderung erst Utilities zu laden, zum anderen müßte dann jeder, der dieses Programm benutzen will, ebenfalls diese Spracherweiterung haben. Die Retter in der Not sind hier Peek und Poke. Diese beiden Befehle greifen direkt auf den Inhalt bestimmter Speicherzellen zu. Auf dem Verändern von Werten in Speicherzellen basiert letztlich die ganze Arbeit des Computers. Wenn man also die Bildschirmfarbe Schwarz haben will, muß man das nur dem Videochip mitteilen. Wie beim Briefeverschicken muß jedoch die Adresse stimmen. In diesem Fall wäre sie 53281. Wird der Wert in dieser Adresse geändert, wechselt der Commodore prompt die Farbe. Bei all diesen Peeks und Pokes gibt es aber auch ein Problem. Was zuerst wie ein Riesenvorteil wirkt, die reichhaltige Auswahl, wird spätestens dann problematisch, wenn man ein Programm länger nicht mehr gesehen hat, es dann wieder listet und sich schließlich den Kopf darüber zerbrechen muß, was denn nun dieser Poke oder jener SYS-Aufruf bedeutet. Wer wird sich all diese Adressen merken? Sie auswendig zu lernen, ist ähnlich ratsam, wie in China Reiskörner zu zählen. Da sollte man sein Gedächtnis besser für sinnvollere Anwendungen nutzen. Es gibt natürlich auch irgendwo Aufzeichnungen darüber, welche Poke-Adressen welche Bedeutung haben. Leider sind die aber meistens weder übersichtlich, noch geordnet. Deshalb wächst sich das Programmieren mit Pokes meist zu einem endlosen Forsten in diversen Heften, Büchern und anderen Nachschlagewerken aus. So wühlt man dann ▶



stundenlang, immer wieder denselben Spruch vor sich hinmurmelnd: „Aber ich hatte diesen Poke doch erst vor ein paar Tagen irgendwo . . ." Diesem Zustand wollen wir abhelfen: Dazu haben wir so ziemlich alles, was an wichtigen und interessanten Peeks und Pokes im Speicher des Commodore versteckt ist, zusammengekramt und auf zwei Poster verteilt.

Poster zeigen Weg durchs wilde Pokeistan

Zusätzlich zu den Adressen gibt es noch knappe Erklärungen und, wenn nötig, kleine Beispiele. In diesem und im nächsten Heft werden Sie jeweils ein Poster finden — als Wegweiser durch den Speicherdschungel. Wenn Sie es in der Nähe Ihres Computers aufhängen, dann sollte im Normalfall ein Blick ausreichen, um eine gewünschte Adresse zu finden.
Nur die Erklärungen auf dem Poster sind sehr knapp. Und weil der Heftpreis sehr darunter litte, wenn wir eine Lupe beilegten, haben wir uns gegen einen noch kleineren Druck entschieden. Zunächst werden wir die Poster herausbringen, damit man schon mal was in der Hand und an der Wand hat. In den folgenden Heften finden Sie Artikel, die überall da, wo es ratsam erscheint, ausführlichere Erklärungen liefern. Über Peeks und Pokes lernt man am meisten immer noch durch ausprobieren. Wenn Sie also etwas nicht verstehen, dann versuchen Sie es einfach. Lesen Sie dann noch einmal die Erklärung, müßte eigentlich alles klar sein. Wir setzen dabei Kenntnisse über Bits und Bytes voraus, die nicht jedem bekannt sein werden. Um dem Abhilfe zu schaffen, erklären wir die Begriffe Bit, Byte und Adresse kurz noch einmal.

Bits und Bytes

Um sie dreht sich alles im Computer. Ein Bit ist die kleinste Einheit bei der Informationsspeicherung. Es kann den Wert 1 oder 0 haben. Der Commodore 64 ist ein 8-Bit-Rechner. Und dieser Name verrät es auch schon: Jeweils 8 Bits werden zu einem „Byte" zusammengefaßt. Und das hat dann eine „Adresse". Diese Adresse kann beim 64er von 0 bis 65535 reichen. Allerdings ist ein Teil dieser Adressen vom ROM-Speicher belegt. Aber dazu später. Um die ganze Sache einfacher zu machen, werden diese acht Bits zu einer Dualzahl zusammengefaßt: zum Beispiel 01101011. Die Dualzahlen setzen sich immer nur aus den Zahlen 0 und 1 zusammen. Jedes Bit in einer Dualzahl hat neben dem Wert 0 oder 1 noch eine eigene Nummer. Sie bestimmt sich aus der Position des jeweiligen Bits innerhalb der Dualzahl. Es wird von rechts nach links durchgezählt und mit 0 angefangen. Das Bit links außen hat die Nummer 7, das Bit rechts außen die Nummer 0. Will man nun den Wert eines Bytes wissen, sucht man die Bits mit dem Wert 1 heraus. Sie sind gesetzt. Die Nummer eines gesetzten Bits ergibt den Exponent zur — gleichbleibenden — Basis 2. Und endlich der letzte Schritt: Die Zweierpotenzen aller gesetzten Bits werden addiert und ergeben den Wert des Bytes. Also ist

$00000011 = 2^1 + 2^0 = 2 + 1 = 3$.
Oder $01101011 = 2^6 + 2^5 + 2^3 + 2^1 + 2^0 = 64 + 32 + 8 + 2 + 1 = 107$.

Soviel zu Bit und Byte. Jetzt wollen wir aber die einzelnen Adressen vom Poster, hier noch mal genauer unter die Lupe nehmen.

Die einzelnen Adressen

Die Speicherzelle 1 ist eine der wichtigsten überhaupt. Denn der Commodore 64 hat ja viel mehr Bausteine eingebaut, als es sein Adressbereich (0-65535) überhaupt erlauben würde. Alles in allem befinden sich im 64er über 86 KByte. Um zwischen den verschiedenen Bausteinen umschalten zu können, gibt es die Adresse Nummer 1. Die Programmiersprache Basic befindet sich zum Beispiel in 8K-ROM (Nur-Lese-Speicher) ab der Adresse 40960. Man kann dieses ROM aber ausblenden und an derselben Stelle im Speicher-RAM (also Schreib-/Lesespeicher) haben. Und dort kann sich dann zum Beispiel ein Maschinenspracheprogramm oder eine andere Programmiersprache befinden. Wenn Sie mit dieser Technik ein wenig experimentieren wollen, probieren Sie folgende Zeile:
FOR X = 40960 TO 49152: POKE X, PEEK(X): NEXT X: POKE 1,54
Dieser Ausdruck sorgt dafür, daß der Inhalt des Basic-ROMs ins RAM kopiert wird, das an der gleichen Adresse liegt und dann mit Poke 1,54 gewählt wird. Die ganze Aktion funktioniert deshalb so einfach, weil der Commodore 64 folgende — nützliche — Eigenheiten hat: Der Peek-Befehl, der einen Wert aus einer Speicherzelle ausliest, macht das immer mit der gewählten Adresse. Also aus dem ROM, wenn es aktiv ist, andernfalls eben aus dem RAM. Der Poke-Befehl hingegen wirkt im Zweifelsfall immer auf das RAM.

Kopieren des ROM ins RAM

Das ist auch sehr vernünftig von ihm. Denn Poke schreibt einen Wert in eine Speicherzelle und ins ROM könnte er das ja nicht.
Die Zeile oben liest also einen Wert aus dem ROM und poked ihn ins RAM auf der gleichen Adresse. So wird mit ganz einfachen Mitteln das ROM ins RAM kopiert. Zuletzt wird dann noch das RAM aktiviert. Denn Bit-Nummer 0 ist dafür zuständig, ob im Bereich 40960 bis 49152, dem Basic-Bereich, das ROM oder das RAM aktiviert wird. ROM ist gewählt, wenn das Bit an ist, RAM wenns aus ist. Der Normalwert dieser Adresse ist 55. Wenn das letzte Bit aus ist, haben wir eben 54. Vorsicht, wenn Sie andere Werte haben oder vergessen, das Basic zu kopieren. Falls die Pro-

grammiersprache und/oder das Betriebssystem ausgeblendet werden, ohne daß für Ersatz gesorgt ist, kann das recht merkwürdige Konsequenzen haben.
Das Bit 1 ist für einen anderen Speicherbedarf zuständig: das Betriebssytem. Es liegt von 57334 bis 65535. Aber jetzt kommen wir gleich zu einer weiteren Eigenart des Commodore 64. Schwer zu verstehen, warum nicht das Betriebssytem allein ins RAM gelegt werden kann. Zumindest geht dies nicht von Basic aus. Also ist Bit 1 streng genommen für Basic und Betriebssytem gleichzeitig zuständig.

Bit 2 für Zeichensatz

Bit 2 schließlich ist für den Zeichensatz verantwortlich. Der liegt nämlich an derselben Adresse, wie die sogenannten I/O-Register. Letztere sind Adressen, die direkt auf irgendwelche Ein-/Ausgabebausteine wirken, wie zum Beispiel der Farb-Poke 53281. Unter derselben Adresse ist auch das ROM zu erreichen, in dem das Aussehen der Zeichen gespeichert ist. Normalerweise liegt es da gut und stört niemanden. Denn der Grafikchip ist der einzige, der dieses ROM ständig braucht. Uns als Benutzer interessiert normalerweise nicht, welche Daten in diesem Baustein stecken. Mit einer Ausnahme: Wenn wir die Zeichen selbst verändern wollen. Doch bleiben wir bei der Speicheraufteilung. Die Adressen 43 und 44 sollen uns als nächste interessieren. Sie legen fest, wo im RAM das Programm anfängt, normalerweise ab Adresse 2048. Zahlen, die größer als 255 sind, müssen in zwei Bytes zerlegt werden. Man spricht von High Byte (HB) und Low Byte (LB). Dazu dient folgende Formel:

LB = (Adresse-(INT(Adresse/256) * 256))
HP = INT (Adresse/256)

Wenn Sie also den Anfang Ihrer Basic-Programme verlegen, setzen Sie einfach die neue Startadresse in diese Formeln ein und poken das Low Byte in (43), das High Byte in (44). Halt! Das war's noch nicht ganz. Um ab dieser neuen Adresse ein Basic-Programm abspeichern zu können, müssen Sie noch in die neue Startadresse eine 0 poken und ein NEW ausführen, damit der neue Speicher auch gelöscht wird. Das Ende eines Programms werden Sie selten selbst angeben. Es ergibt sich ja automatisch aus den Zeilen des Basic-Programms. Interessiert Sie später, bis zu welcher Adresse Ihr Programm geht, erfahren Sie es mit dem Befehl: PRINT PEEK (45) + 256 ★ PEEK (46)
Nach demselben Prinzip arbeiten auch die beiden nächsten Speicherzellen: 55/56 sind dafür zuständig, wo das RAM des Arbeitsspeichers endet. Wenn man ihm sagt, das RAM ist 8K früher zu Ende, dann akzeptiert er das. Und geht ab sofort von weniger Speicher aus. Welchen praktischen Nutzen das hat? Zum Beispiel können Sie Maschinenprogramme oder Grafiken in diesem, so „geschützten" Speicherbereich ablegen. Versuchen Sie doch mal Poke 56,80. Nach Print FRE(0) werden Sie überrascht sein, was der Commodore 64 aus seinen 38K-RAM gemacht hat. Der Normalwert ist übrigens Poke 56, 160. Verlassen wir nun aber diese Speicherangelegenheiten und schauen wir uns in der Umwelt des Computers um. Gemeint sind die Zusätzgeräte.

Peek und Peripherie

Mit Peek (186) können Sie ermitteln, welches Zusatzgerät gerade aktiv war/ist. Wenn Sie einen Kassettenrecorder haben, ist das meist Nummer 1. Besitzen Sie eine Floppy, so steht hier 8. Und wenn Sie gar mit einem Drucker arbeiten, finden Sie die 4.

Die nächste Adresse ist der Zähler für den Tastaturpuffer. So etwas hat der Commodore nämlich auch. Er ist eigentlich dafür gedacht, daß niemand beim Schnelltippen den Commodore überholt. Deshalb können bis zu zehn Tastenanschläge zwischengespeichert werden, bevor sie der Computer ausführt. Der Inhalt der Adresse (198) gibt nun an, wieviele Zeichen sich bereits in diesem Tastaturpuffer befinden. Sie können sich das zunutze machen, indem Sie beispielsweise abfragen, ob jemand irgendeine Taste gedrückt hat, während Ihr Programm lief. Eine weitere Anwendung ist die simulierte Tastatur. Sie können den Inhalt dieses Puffers durch ein Programm ändern und so bis zu zehn Tasten „vorprogrammieren". Welche Vorteile das hat, und wie's geht, verraten wir im nächsten Heft.

Christian Spanik

DURCHS WILDE POKEISTAN

Ein Wegweiser durch den Speicherdschungel Teil II

Da sind wir wieder. Weiter geht es mit unserer Reise durch den Speicherdschungel des Commodore 64. Gelandet waren wir nicht im Urwald, sondern bei Adresse 198. Jetzt starten wir mit 199.

Wenn sich jemand fragt, warum wir uns solange in den niedrigen Speicherbereichen aufhalten, noch eine kurze Erklärung vorher. Wer eh' schon alles über ROM, RAM und Betriebssystem weiß, kann diesen Teil getrost überspringen.

[illegible] – fein, daß noch einer hiergeblieben ist. Nun also zur Erklärung:

Alles, was der Commodore 64 kann (oder auch nicht kann . . .), steht in irgendwelchen Speicherzellen. Und damit er's auch nicht vergißt, wenn der Strom ausgeschaltet wird, steht es in einem bestimmten Speicherbereich: im sogenannten ROM-Speicher. ROM heißt Read Only Memory. Dieser Speicher hat einen Vorteil: Was hier steht, das bleibt auch da. Fein, sagt man sich. Problem gelöst. Denn selbst ohne Strom sind die Daten hier sicher wie in Abrahams Schoß. Aber die Daten, von denen wir gerade sprechen, stellen das sogenannte Betriebssystem dar. Wer will, kann auch Kernal dazu sagen. Im allgemeinen kann uns das ziemlich wurscht sein. Aber eines sollte man doch darüber wissen: Ohne Betriebssystem kein Basic, ohne Basic keine selbstgeschriebenen Programme und ohne selbstgeschriebene Programme weder Spaß noch Erfolgserlebnisse. Anders ausgedrückt: Ohne Betriebssystem ist der Computer so sensationell, wie ein Reiskorn auf Chinas Reisfeldern.

Das Betriebssystem ist ein äußerst nützliches Programm, das ständig am Laufen ist. Wer mitgedacht hat, dem wird jetzt schon eine ganz bestimmte Frage auf der Zunge liegen. Und damit wir Ihre Zunge nicht allzusehr beanspruchen, wollen wir die Frage laut stellen: Wenn das Betriebssystem ein Programm ist, und es gleichzeitig in einem ROM-Speicher steht, den man softwaremäßig nicht verlängern kann, wie kommt es dann bitteschön, daß so ein Programm funktioniert? Eine FOR-NEXT-Schleife zum Beispiel kann doch in einem festen Speicher nicht realisiert werden.

Und genau das ist das hüpfende Komma. Jedes Programm braucht immer wieder Speicherzellen, die ▶



veränderbar sein müssen, in denen Werte abgelegt werden können. Auch das Betriebssystem kommt da nicht drumherum. Um ein einfaches Beispiel zu nehmen: Die höchste und die niedrigste verfügbare RAM-Speicheradresse müssen dem Computer nach dem Einschalten klar sein.
Sonst würden wir buchstäblich im Urwald stehen.
Damit das nicht passiert, werden die beiden Werte im ROM festgelegt. Andererseits lieben es Programmierer, ausgerechnet diese Werte zu verändern. Die Gründe

Betriebssystem

dafür sind vielfältig und erst einmal unwichtig. Um das Ändern möglich zu machen, müßte man die Werte im RAM verfügbar haben. Ganz so einfach ist es leider nicht. Denn es nützt nichts, wenn wir in irgendeiner RAM-Speicheradresse neue Werte festlegen, der Computer aber seine Werte immer noch aus dem ROM holt. Die Lösung des Dilemmas heißt Kopieren. Und zwar ein ganzes Eck des Betriebssystems in den freien RAM-Bereich. Und genau das macht der Commodore 64 jedesmal beim Einschalten. Bestimmt ist dem einen oder anderen aufgefallen, daß es einen Moment dauert, bis die Einschaltmeldung auf dem Bildschirm erscheint. Das liegt daran, daß der Commodore jedesmal beim Einschalten erst initialisiert. Er stellt die Grundfarben des Bildschirms ein (dieses herrliche Blau), die Schriftfarbe (das sich so herrlich kontrastreich vom erwähnten Dunkelblau absetzende Hellblau) und kopiert eben auch (ganz legal!) Teile des Betriebssystems in den RAM-Bereich. Und weil das der Commodore selber macht, holt er sich seine Informationen in Zukunft aus dem RAM. Praktisch, oder? Bestimmte Grundwerte sind sicher vor dem Ausschalten im ROM. Nach dem Einschalten bleiben sie veränderbar im RAM.
Hier zeigt sich, daß der Commodore 64 zwar ein 64-K-RAM-Computer ist, aber dem BASIC-Programmierer nur 38 911 sogenannte BASIC-Bytes zur Verfügung stellt. Warum Commodore dann trotzdem von 64 K-RAM freiem Programmierspeicher redet? Erstens, weil man tatsächlich fast soviel Platz hat, wenn man in Maschinensprache programmiert und zweitens . . . also . . . es klingt ja auch ganz gut, oder?
Wir hoffen, daß jetzt jedem klar ist, warum dieser niedrige Speicherbereich, die sogenannte Zeropage, so wichtig ist. Dort befinden sich die Informationen des Betriebssystems. Deshalb sollte man vermeiden, hier wild herumzupoken, sonst kannein Programm verloren gehen. Und das ist niemandem zu wünschen.
Nachdem jetzt wohl alle Klarheiten beseitigt wären, können wir mit unserer Speicherzelle 199 weitermachen.
Sie ist ein sogenannter Flag. Oder auch ein Schalter, ein softwaremäßiger Schalter allerdings. Denn nur ein solcher Schalter hält das Ein-und Ausschalten hundertmal in der Sekunde aus.
Wenn dieser Schalter angeknipst und das zuständige Bit auf 1 gesetzt wird, aktiviert das einen bestimmten Zustand. Im Fall der Adresse 199 schaltet man damit den Reverse-Modus an. Die Textausgabe schaltet auf invertierte Zeichen um. Dafür gibt es auch eine entsprechende Taste: die CTRL-9, für RVS ON. Das Bit 1 bei 199 hat dieselbe Funktion. Mit POKE 199,1 wird dieser Modus angeschaltet. Der POKE hat bei

Speicherzelle 199

Listings gegenüber dem Steuerzeichen den Vorteil, daß er nicht verwechselt werden kann. Mit POKE 199,0 kann der Reverse-Modus wieder abgeschaltet werden.

Ein andere wichtige Adresse ist 203. Hier kann man mit PEEK (203) ablesen, welche Taste gerade gedrückt wird. Leider haben die Werte in dieser Adresse weder Gemeinsamkeiten mit den ASCII- noch mit den Bildschirmcodes. Es sind ganz eigene Zahlen. Am besten legt man sich eine Liste an, die darüber Auskunft gibt, welche Taste zu welcher Zahl in der Speicherzelle 203 gehört (siehe RUN 6). Im Normalfall — wenn keine Taste gedrückt wird — steht hier 64.

Adresse 204

Die Adresse 204 ist wieder ein Flag: der sogenannte Cursorflag. Wie der Name schon vermuten läßt, hat dieser Flag irgendwas mit dem Cursor zu tun. Um Licht in die ganze Sache zu bringen, wollen wir auch gleich verraten, was das für ein Zusammenhang ist. Dieser Flag ist dafür zuständig, ob der Cursor erscheinen soll oder nicht. Nicht mit der Blinkphase des Cursors verwechseln! Die Speicherzelle 204 entscheidet darüber, ob der Cursor überhaupt zu sehen ist. Während ein Programm läuft, sieht man diese kleine Schreibmarke nicht. Erst wenn eine INPUT-Anweisung im Programm vorkommt, oder wenn sich der Commodore 64 im Direktmodus befindet, ist der Cursor plötzlich wieder da.

Nun wäre es praktischer, wenn er auch während eines laufenden Programmes auftauchen würde. Zum Beispiel, um bei einem Textverarbeitungsprogramm die aktuelle Schreibposition anzuzeigen. Der Inhalt dieser Adresse muß 0 sein, damit der Cursor angeschaltet ist, und 1, damit er aus bleibt. So einfach ist das. Am besten, man probiert diesen Poke einfach aus. Das gilt genauso für andere Adressen. Aber Achtung: der Cursor steht nicht in der Landschaft herum wie ein Tourist vor dem Eiffelturm, sondern er blinkt munter vor sich hin. Und damit er schön im Takt bleibt, gibt es einen Zähler in der Adresse 207. Dort läuft ständig ein Countdown ab. Im Gegensatz zum Space-Shuttle-Programm ist es bei unserem Blinkzähler so, daß der Cursor jedesmal bei einem bestimmten Wert, wie geplant, verschwindet

oder erscheint. Mit der Adresse 207 selbst kann man nun allerdings nicht so wahnsinnig viel anfangen. Denn hier wird so schnell gezählt, daß man von BASIC aus gar keinen Einfluß darauf nehmen kann. Warum wir dann die ganze Zeit davon erzählen? Weil wir Sie auf folgende Kleinigkeit aufmerksam machen wollen: Wenn der Cursor „programmiert", also wie vorhin erklärt, eingeschaltet wurde, dann kann es beim Ausschalten passieren, daß er sich weigert, zu verschwinden. Das hat aber nichts damit zu tun, daß im Commodore 64 ein geheimes, eigenwilliges Leben existiert, wie uns das Hollywood in TRON zu erklären versucht. Man kann es auch nicht irgendeiner fremden Macht in die Schuhe schieben, sondern es ist einfach so, daß sich das Betriebssystem nach dem Abschalten des Cursors gar nicht mehr um seinen weiteren Verbleib kümmert.
Und so steht dann plötzlich ein einsamer, stiller Cursorblock irgendwo auf dem Bildschirm.
Wenn Sie aber beim Ausschalten auch gleichzeitig den Blinkzähler auf 0 setzen, dann ist der Cursor auf jeden Fall aus, bevor er abgeschaltet wird. Wie sich das gehört. So, und weil wir jetzt wissen, wie man den Cursor ein- und ausschaltet, wollen wir gleich noch festlegen, wo er erscheinen soll. Dazu müssen Sie die X- und Y-Koordinaten auf dem Bildschirm kennen. Der Bildschirm besteht horizontal aus 40 Zeichen und vertikal aus 25 Zeilen. Da Computer aber

Cursor-Steuerung

beim Zählen (fast) immer bei 0 anfangen, muß der X-Wert, der die horizontale Lage des Cursors angibt, zwischen 0 und 39 liegen. Und der Y-Wert zwischen 0 und 24. Jetzt müssen Sie nur noch wissen, wohin mit diesen Werten, und schon können Sie den Cursor auf dem Bildschirm herumjagen. Die nötigen Adressen sind 211 für X-Wert und 214 für Y. Wenn der Cursor also in der 4. Zeile und der 10. Spalte auftauchen soll, poken Sie 211,4 und 214,10. Weil aber der Commodore dazu neigt, solche Eingriffe in seine interne Verwaltung zu ignorieren, muß noch eine Routine des Betriebssystems aufgerufen werden, die die neuen Werte verarbeitet. Dazu reicht ein einfaches SYS 58 732.
Wenn Sie das gemacht haben — egal an welcher Stelle des Programms —, dann erfolgt die nächste PRINT-Ausgabe am gewünschten Platz.
Die nächste Adresse ist schon etwas für die Profis. Wenn Sie also das alles nicht gleich verstehen, machen Sie sich nichts daraus. Wenn erstmal mehr Erfahrung mit dem Commodore vorhanden ist, dann versteht man plötzlich Dinge, die einem vorher völlig unklar waren. Bei der Gelegenheit gleich noch ein Tip an alle Anfänger: Auch bei Artikeln, die zuerst keinen großen Sinn zu machen scheinen, sollte man möglichst versu-

Tastaturpuffer

chen, zu verstehen, um welches Problem es geht. Irgendwann kommt wahrscheinlich der Moment, wo man vor genau denselben Schwierigkeiten steht. Und dann wird einem auch plötzlich klar, was in dem Artikel steht, und wie es gemeint ist. Naja, zumindest meistens . . .
Jetzt aber zurück zum Thema: Wir haben bereits im letzten Teil den Tastaturpuffer erwähnt. Das ist ein 10-Byte-großer Bereich, in dem die Codes der Tasten abgelegt werden, die nicht sofort verarbeitet werden können. Zu diesem Puffer gehören einige Speicherzellen, die man ausnutzen kann. In der Adresse 198 steht zum Beispiel, wieviele Tasten bereits im Tastaturpuffer liegen. Gemeint ist, wieviele verschiedene Codes bereits hier sind.
Wenn das Keyboard im Puffer verschwinden würde, hätten wir nichts mehr zu tippen.

Auf jeden Fall kann man sich diese Adresse zunutze machen, wenn man die simulierte Tastatur anwendet. Nein, das ist kein neues Kartenkunststück, sondern ein sehr nützliches Instrument. Funktionieren tut es so: Viele Dinge beim Commodore lassen sich nur im Direktmodus erledigen. Oder es ist besser, wenn man sie direkt eingibt.
Ein Paradebeispiel ist der Befehl LOAD. Wenn Sie einzelne Programme miteinander verketten wollen, so daß ein Teil den anderen nachlädt, ist allerhand zu beachten. Ist das nachgeladene Programm sehr viel größer als das letzte, dann kann es passieren, daß der überstehende Teil kurzerhand verschwindet. Denn alle Zeiger, die auf Programmende, Variablenliste etc. zeigen, stimmen dann nicht mehr.
Wie der Fachmann diese Erscheinung nennt, wissen wir leider nicht. Die meisten Programmierer haben nur ein Schmipfwort dafür, das sich aber beim besten Willen nicht als Spezialausdruck verkaufen läßt. Also verzichten wir darauf.
Dafür wissen wir aber, wie man dazu sagt, wenn man den Computer trotzdem überlistet: Overlay. Aber es geht auch einfacher. Man schreibe einfach LOAD „Programmname" und RUN auf den Bildschirm und lege hinter jeden Ausdruck eine RETURN-Taste im Puffer ab, verlasse den Programm-Modus und erfreue sich daran, das jetzt alles wie von Geisterhand funktioniert.
Um es ausprobieren zu können, müssen Sie noch die Adresse des Tastaturspeichers wissen. Er beginnt bei 631. Einfach die Codes der entsprechenden Buchstaben hierhineinpoken. Danach die Anzahl der Tasten in Adresse 198 poken und den Computer in den Direktmodus bringen. Zum Beispiel

Adresse 631

durch ein END. Auf dem Peek und Poke Poster findet sich unter der Adresse 631 ein kleines Beispielprogramm, das den Befehl LIST in den Puffer bringt. Es werden einfach die ASCII-Werte für die Buchstaben und eine 13 (für RETURN) in die Speicherzellen 631 bis 636 gepoked. Nachdem danach auch das Programm selbst zu Ende ist, begibt sich das Betriebssystem auf die Suche nach neuer Arbeit und stößt dabei auf die Werte im Tastaturpuffer. Und weil er eh' gerade nichts besseres zu tun hat, führt es diese Befehle aus.

Alles klar? Jetzt wieder zu einfacheren Anwendungen. ▶



Die Adresse 646 ist für die jeweilige Schriftfarbe zuständig. Genauso, wie sich die Farbe mit den Tasten CTRL oder C= (Commodore-Taste) ändern lassen, geht dies auch mit einem Poke. Oder Farbenblinde und Besitzer von Schwarzweiß-Fernsehern können mit einem Peek herausfinden, was für eine Schriftfarbe gerade verwendet wird. Der Nutzen solcher Adressen liegt in erster Linie darin, daß man sie statt Steuerzeichen verwenden kann, die leicht verwechselt werden können.

Noch eine letzte Adresse zum Thema Tastaturpuffer. Wir haben ja schon gesagt, daß er zehn Zeichen aufnehmen kann. In einigen Fällen kann es wünschenswert sein, diesen Puffer zu verkleinern. Zum Beispiel, wenn im Verlauf eines Spieles bestimmte Tasten zur Steuerung verwendet werden und dann später ein INPUT erfolgt. Normalerweise würden die zuletzt gedrückten Tasten auf dem Schirm erscheinen. Das ist aber selten im Sinne des Erfinders. Wenn Sie den Tastaturpuffer abschalten oder entsprechend verkleinern, dann kann das nicht passieren. Achtung: nicht beim Direktmodus abschalten. Denn ohne Tastaturpuffer kann der Commodore keine Eingabe mehr annehmen.

Adresse 649

Die maximale Größe des Tastaturpuffers ist in der Adresse 649 festgelegt. Hier steht normalerweise der Wert 10. Wetten, daß irgendwelche Spaßvögel jetzt auf die Idee kommen, den Puffer auf 255 auszudehnen. Das ist der höchste Wert, den man in eine Speicherzelle poken kann. Nur sind wir immer noch in der Zeropage. Und die Werte hier sind sehr allergisch gegen das Überschreiben. Dem Betriebssystem würde es auch nicht gefallen, wenn es statt der gewohnten Daten plötzlich irgendwelche ASCII-Codes vorfindet.

Und die Moral von der Geschicht', Puffer vergrößern gehört sich nicht.

Ein paar Worte noch zum Unterschied von 198 und 649. Die beiden scheinen sich ja ziemlich ähnlich zu sein. Aber der Unterschied ist ganz einfach: 649 gibt die maximale Größe des Speichers an. 198 dagegen zeigt an, wieviel tatsächlich drin ist.

Apropos Tasten. Jeder hat schon mal festgestellt, daß es am Commodore 64 einige Tasten gibt, die eine automatische Wiederholfunktion haben. Drückt man zum Beispiel auf die Cursortaste nach oben, dann flitzt der Cursor solange in diese Richtung, bis er gegen den oberen Bildschirmrand donnert. Nur bei normalen Buchstaben geht das nicht. Denkste. Geht doch. Die Adresse 650 steuert diese Funktion. Es gibt drei mögliche Einstellungen: Poked man 0 in die Adresse, so gilt die normale Belegung. Bei 64 wird überhaupt keine Taste automatisch wiederholt. Und bei 128 funktioniert's plötzlich bei allen.

Zeichensätze

Ein anderes Thema: die beiden Zeichensätze. Der Commodore 64 hat zwei davon. Der eine ist für Texte interessant, weil er Groß- und Kleinschreibung möglich macht. Der andere ist der Grafikzeichensatz. Es kann immer nur einer aktiv sein. Deshalb kann man mit der Tastenkombination SHIFT und C= zwischen diesen beiden umschalten. Das kann aber auch unangenehme Folgen haben: Textdarstellungen werden unleserlich und aus einer tollen Grafik wird plötzlich ein Buchstabensalat. Es gibt mehrere Arten, diese Umschaltung zu blockieren. Entweder mit dem Printen von CHR$(8) oder mit Pokes. Verwendet wird die Adresse 657. Mit POKE 657,128 erreichen Sie eine Verriegelung der Umschaltung. Mit POKE 657,0 kann man das auch wieder rückgängig machen. Jetzt kommt noch schnell die Sprungadresse des USR-Befehls.

Sprungadresse

Mit PRINT USR(X) kann ein eigenes Maschinenprogramm aufgerufen werden. Genau wie mit SYS. Nur besteht ein Unterschied: Der USR-Befehl ist eine Möglichkeit, eigene Befehslerweiterungen zu programmieren. Das X bei USR(X) ist ein Wert, der einem Maschinenprogramm übergeben werden kann. Genaueres über diese Technik empfehlen wir aber aus Büchern zu entnehmen, die sich mit der Programmierung in Maschinensprache befassen. Wenn wir hier ins Detail gehen, dann kommen wir wirklich in den Urwald. Und wir sind doch gerade dabei einen möglichst gangbaren Weg durchs wilde Pokeistan zu finden.

Zuletzt noch zum Kassettenpuffer. Er liegt von Adresse 828 bis 1019 und hat folgende Aufgabe: Bei einem LOAD oder SAVE mit der Datasette ist der Datenaustausch sehr langsam. Damit bei der Übergabe der Daten keine langen Wartezeiten entstehen, werden die Bytes auf dem Weg von oder zur Kassette zwischengespeichert.

Kassettenpuffer

Wie ein Stoßdämpfer nimmt dieser Speicher erst 192 Bytes auf, und sendet diese dann weiter. Wer meint, was geht's mich an, wie der Computer intern mit seinen Daten rumwurschtelt, irrt. Es ist auch für den Programmierer interessant. Denn solange kein Datenaustausch stattfindet, ist dieser Bereich frei. Da kann man zum Beispiel hübsche Spritemuster ablegen oder nützliche Maschinenroutinen. Sobald wieder Daten mit der Datasette ausgetauscht werden, gehen sie verloren. Der Umgang mit der Floppy hingegen macht diesem Bereich gar nichts aus.

Mit Bildschirm, RAM und Anhang geht's nächstes Mal weiter. Bis dann also. Und gut Poke.

Christian Spanik
Hannes Rügheimer